KIET GROUP OF INSTITUTIONS

AI (MSE -1)

Project name – N Queen problem

Name – Shikha Tomar

Roll no. – 2024011004002024

Branch – CSE AI-ML

Section - C

N- QUEEN PROBLEM

**The N-
Queens problem is a classic puzzle in computer science and mathematics. It involves placing N chess queens on an N×N chessboard so that no two queens threaten each other. This means no two queens can be in the same row, column,**

METHODOLOGY

Define : - Board: An N*N grid

Queen: Pieces that can attack vertically, horizontally and diagonally.

Objective:

Place N queen on the board so that none of them attack each other.

Constraints:

Only one queen per row, column and diagonal.

Solution Techniques:

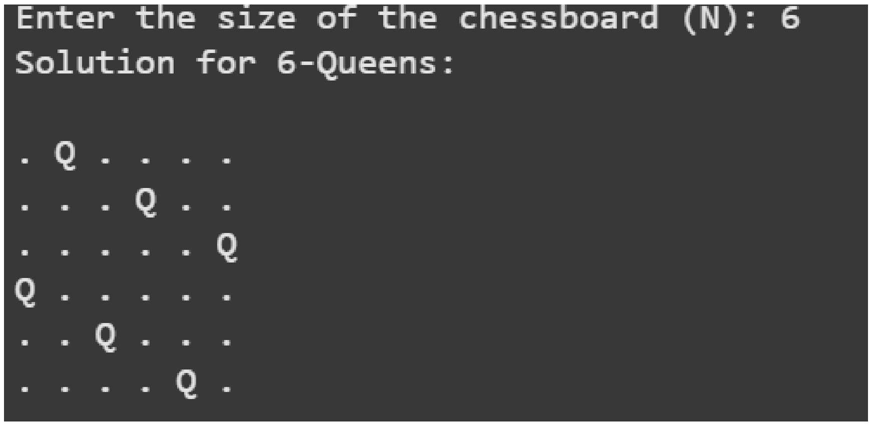
1) Backtracking :- A common method where queens are placed one by one in different columns, and for each columns it checks if the queen can be placed in that column without conflicts.

CODE

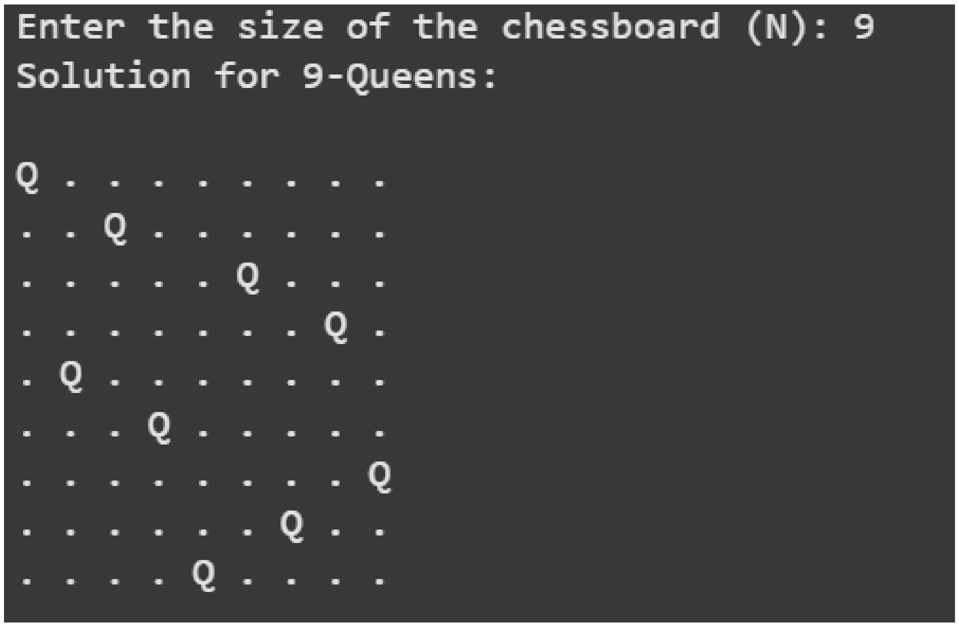
```
# Function to check if placing a queen at
board[row][col] is safe
def is_safe(board, row, col, n):
    # Check the same column
    for i in range(row):
        if board[i][col] == 1:
            return False

    # Check upper left diagonal
    for i, j in zip(range(row, -1, -1),
range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    # Check upper right diagonal
    for i, j in zip(range(row, -1, -1),
range(col, n)):
        if board[i][j] == 1:
            return False

    return True

# Function to solve the N-Queens problem
using backtracking
def solve_n_queens(board, row, n):
    # Base case: If all queens are placed
    if row >= n:
        return True

    # Try placing a queen in each column of
the current row
    for col in range(n):
        if is_safe(board, row, col, n):
            # Place the queen
            board[row][col] = 1

            # Recur to place the rest of the
queens
            if solve_n_queens(board, row + 1,
n):
                return True

            # If placing queen in
board[row][col] doesn't lead to a solution,
backtrack
            board[row][col] = 0

    return False

# Function to print the chessboard
def print_board(board):
    for row in board:
        print(" ".join("Q" if col == 1 else
"." for col in row))
```

```python
    print("\n")

# Main function to solve the problem
def n_queens(n):
    # Create an empty chessboard
    board = [[0] * n for _ in range(n)]

    if solve_n_queens(board, 0, n):
        print(f"Solution for {n}-Queens:\n")
        print_board(board)
    else:
        print("No solution exists.")

# Input the size of the chessboard
n = int(input("Enter the size of the
chessboard (N): "))
n_queens(n)
```

output



```
Enter the size of the chessboard (N): 6
Solution for 6-Queens:

. Q . . . .
. . . Q . .
. . . . . Q
Q . . . . .
. . Q . . .
. . . . Q .
```



```
Enter the size of the chessboard (N): 9
Solution for 9-Queens:

Q . . . . . . . .
. . Q . . . . . .
. . . . . Q . . .
. . . . . . . Q .
. Q . . . . . . .
. . . Q . . . . .
. . . . . . . . Q
. . . . . . Q . .
. . . . Q . . . .
```